WAIT: Anmeldung mit Active-Directory-Konten einrichten

Ausführliche Schritt-für-Schritt-Anleitung, um WAIT (`I3MC-eu/WAIT`) an einen Windows-Domänencontroller anzubinden. Danach melden sich Schüler und Lehrer mit ihrem normalen Schulkonto an.

Die Beispielwerte in dieser Anleitung sind:

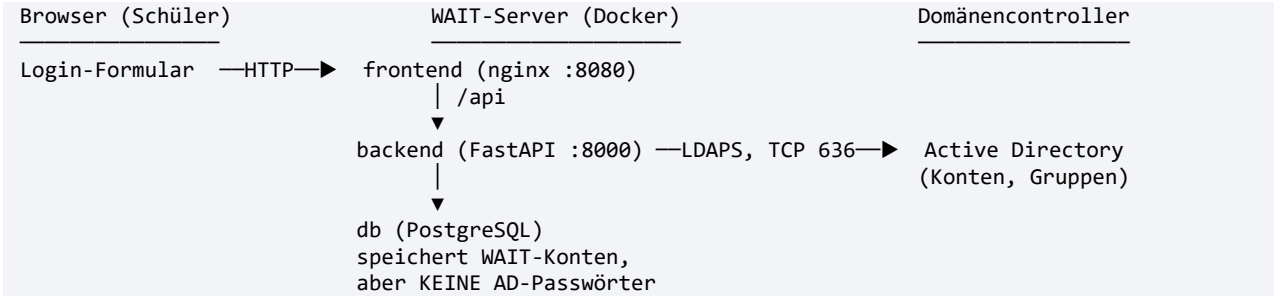
Was	Beispielwert	Bei dir
AD-Domäne	`schule.local`	_____
Domänencontroller (Name)	`dc01.schule.local`	_____
Domänencontroller (IP)	`192.168.92.10`	_____
WAIT-Server (Debian, Docker)	`192.168.92.100`	_____
Dienstkonto	`svc-wait`	_____
Admin-Gruppe	`WAIT-Admins`	_____

Fülle die rechte Spalte aus, bevor du anfängst. Dann kannst du die Befehle unten einfach anpassen.

Inhalt

0. Überblick: Was du am Ende hast 3
0.1 Die Bausteine und wozu sie da sind 3
0.2 Wie sie beim Login zusammenspielen 4
1. Grundlagen, die du verstehen solltest 4
1.1 Active Directory, Domänencontroller, LDAP 4
1.2 LDAP vs. LDAPS 4
1.3 DN (Distinguished Name), der „Pfad“ eines Objekts 5
1.4 Die Benutzernamen eines AD-Kontos 5
1.5 Bind 5
2. Was WAIT beim Login genau macht 5
2.1 Die drei Modi (AUTH_MODE) 7
3. Vorbereitung auf dem Domänencontroller 7
3.1 Domäne und DC herausfinden 7
3.2 Dienstkonto anlegen 8
3.3 Admin-Gruppe anlegen 8
3.4 Optional: Gruppe für erlaubte Benutzer 9
3.5 Alle Werte für WAIT auslesen 9
3.6 Einen Testbenutzer prüfen 9
4. LDAPS (verschlüsselte Verbindung) einrichten 9
4.1 Prüfen, ob LDAPS schon läuft 9
4.2 Gibt es eine Zertifizierungsstelle (CA)? 10
4.3 Zertifikat für den DC besorgen 10
4.4 Zertifikat des DC kontrollieren 11
4.5 Das CA-Zertifikat für WAIT exportieren 11
4.6 Firewall 11
5. WAIT-Server vorbereiten 12
5.1 Werkzeuge zum Testen installieren 12
5.2 Netzwerk zum DC prüfen 12
5.3 CA-Zertifikat auf den Server kopieren 12
5.4 Zertifikat des DC vom WAIT-Server aus prüfen 12
6. WAIT konfigurieren 13
6.1 Die Einstellungen im Einzelnen 13
6.2 Wichtig: Hostname statt IP in LDAP_URL 13
6.3 Die `.env` schreiben 14
6.4 Prüfen, ob Compose die Werte sieht 15
6.5 Neu starten 15
7. Testen, Schritt für Schritt 15
Test 1: Dienstkonto + Zertifikat + Suche (ohne WAIT) 15
Test 2: Passwort eines echten Benutzers 15
Test 3: Gruppenprüfung (verschachtelt) 16
Test 4: Health-Check von WAIT 16
Test 5: Login direkt über die API 16
Test 6: Im Browser 16
8. Fehlersuche 16
8.1 Das Backend-Log lesen 16
8.2 Benutzer bekommt „falsches Passwort“, obwohl es stimmt 17
8.3 Benutzer ist kein Admin, obwohl er in der Gruppe ist 17
8.4 Admin lässt sich im Panel nicht zum Kunden machen 17
8.5 Nach `docker compose up -d --build backend` geht gar nichts mehr (502) 18
8.6 „Too many failed attempts“ 18
9. Betrieb: Was du danach wissen musst 18
9.1 Umstellen auf „nur AD“ 18
9.2 Seed-Konten absichern 18
9.3 AD-Kontosperre beachten 18
9.4 Was passiert, wenn … 19
9.5 Ablaufdaten im Blick behalten 19
9.6 Updates von WAIT 19
10. Checkliste 19
Quellen 20

0. Überblick: Was du am Ende hast



- Nur der **Backend-Container** spricht mit dem DC, und zwar ausschließlich verschlüsselt über Port **636**.
- Der Browser des Schülers hat mit dem DC nichts zu tun.
- WAIT speichert keine AD-Passwörter. Es fragt den DC bei jedem Login, ob das Passwort stimmt.

0.1 Die Bausteine und wozu sie da sind

Baustein	Wo	Wozu
Browser	PC des Schülers	Zeigt das Login-Formular. Spricht nur mit WAIT, nie mit dem DC.
Frontend (nginx, Port 8080)	WAIT-Server, Docker	Liefert die Webseite aus und leitet alle `/api`-Anfragen ans Backend weiter.
Backend (FastAPI, Port 8000)	WAIT-Server, Docker	Nimmt den Login an, fragt den DC und stellt das Login-Token aus. Nur das Backend spricht mit dem DC.
Datenbank (PostgreSQL)	WAIT-Server, Docker	Speichert WAIT-Konten (Name, E-Mail, Rolle) und Tickets, aber keine AD-Passwörter.
`.env`	WAIT-Server	Sagt dem Backend, wo der DC ist und mit welchem Konto es suchen darf.
`ca.pem`	WAIT-Server, `certs/`	Damit das Backend dem DC-Zertifikat vertraut. Sonst bricht die Verbindung ab.
Domänencontroller	Windows Server	Kennt alle Konten, Passwörter und Gruppen und beantwortet die Fragen des Backends.
LDAPS (Port 636)	zwischen Backend und DC	Die verschlüsselte Leitung, über die die Fragen laufen.
DC-Zertifikat	auf dem DC	Beweist dem Backend, dass es wirklich mit dem echten DC spricht.
CA (Zertifizierungsstelle)	meist auf dem DC	Hat das DC-Zertifikat ausgestellt. Ihr öffentliches Zertifikat ist die `ca.pem`.
Dienstkonto `svc-wait`	im AD	Damit das Backend überhaupt suchen darf. Das AD erlaubt keine anonymen Suchen.
Gruppe `WAIT-Admins`	im AD	Wer hier drin ist, wird in WAIT Admin.

0.2 Wie sie beim Login zusammenspielen

Nr.	Von → An	Was passiert	Was dafür nötig ist
1	Browser → Frontend → Backend	Der Schüler schickt Name und Passwort an WAIT.	`AUTH_MODE` = `ldap` oder `both`
2	Backend → DC	Verschlüsselte Leitung aufbauen. Der DC zeigt sein Zertifikat, das Backend prüft es.	`LDAP_URL`, `LDAP_CA_CERT_FILE`, DC-Zertifikat, Port 636
3	Backend → DC	Anmeldung als Dienstkonto, damit gesucht werden darf.	`LDAP_BIND_DN`, `LDAP_BIND_PASSWORD`
4	Backend → DC	Den Schüler suchen. Der DC liefert DN, Name und E-Mail.	`LDAP_BASE_DN`
5	Backend → DC	Anmeldung als der Schüler mit seinem Passwort. Klappt das, stimmt das Passwort.	gültiges, aktives AD-Konto
6	Backend → DC	Mit dem Dienstkonto fragen: Ist er in der Admin-Gruppe?	`LDAP_ADMIN_GROUP`, optional `LDAP_USER_GROUP`
7	Backend → Datenbank	WAIT-Konto anlegen oder Name und Rolle aktualisieren.	`LDAP_EMAIL_DOMAIN`, falls das Konto keine E-Mail hat
8	Backend → Browser	Der Browser bekommt ein Token und ist bis zu 24 Stunden eingeloggt. In dieser Zeit wird der DC nicht mehr gefragt.	`JWT_SECRET`

Kurz gesagt: Der Browser redet nur mit WAIT, und nur das WAIT-Backend redet mit dem DC. Der DC beantwortet dabei zwei Fragen: ob das Passwort stimmt und ob der Benutzer Admin ist. Alles, was du einrichtest (Zertifikat, Dienstkonto, Gruppe, `.env`), sorgt dafür, dass diese zwei Fragen sicher gestellt werden können.

1. Grundlagen, die du verstehen solltest

1.1 Active Directory, Domänencontroller, LDAP

- **Active Directory (AD)** ist die Datenbank mit allen Benutzerkonten, Passwörtern, Gruppen und Computern der Schule.
- Der **Domänencontroller (DC)** ist der Windows Server, auf dem diese Datenbank läuft. Gibt es mehrere DCs, haben alle dieselben Daten (Replikation). WAIT braucht nur einen davon.
- **LDAP** ist die „Sprache“, in der andere Programme das AD abfragen. WAIT nutzt sie, um Benutzer zu finden und Passwörter zu prüfen.

1.2 LDAP vs. LDAPS

	Port	Verschlüsselt	In WAIT
LDAP	389	nein, Passwörter gehen im Klartext durchs Netz	nur mit `LDAP_START_TLS=true` (wird dann nachträglich verschlüsselt)
LDAPS	**636**	**ja, TLS ab der ersten Sekunde**	**empfohlen und Standard**

Für LDAPS braucht der DC ein **TLS-Zertifikat**, genau wie eine HTTPS-Webseite. Darum geht es in Kapitel 4.

1.3 DN (Distinguished Name), der „Pfad“ eines Objekts

Jedes Objekt im AD hat einen eindeutigen Pfad, den DN. Beispiel:

```
CN=svc-wait,OU=Service,DC=schule,DC=local
```

Von **rechts nach links** gelesen:

Teil	Bedeutung
`DC=schule,DC=local`	die Domäne `schule.local` (DC = Domain Component, hat nichts mit „Domänencontroller“ zu tun)
`OU=Service`	der Ordner (Organisationseinheit) „Service“
`CN=svc-wait`	das Objekt selbst (Common Name)

Wichtig:

- Der DN muss **exakt** stimmen. Ein Leerzeichen oder ein falscher OU-Name reicht, damit nichts funktioniert. Deshalb liest du DNs in Kapitel 3.4 aus, statt sie zu tippen.
- Groß- und Kleinschreibung ist egal.
- Standardordner wie „Users“ sind **keine** OUs, sondern Container: `CN=Users,DC=schule,DC=local` (CN statt OU).

1.4 Die Benutzernamen eines AD-Kontos

Ein AD-Konto hat mehrere Namen. WAIT akzeptiert beim Login jeden davon:

Attribut	Beispiel	Woher
`sAMAccountName`	`m.muster`	der klassische Anmeldename
`userPrincipalName` (UPN)	`m.muster@schule.local`	Anmeldename im E-Mail-Format
`mail`	`max.muster@schule.de`	E-Mail-Adresse, falls gepflegt
Format `DOMÄNE\name`	`SCHULE\m.muster`	WAIT schneidet `SCHULE\` einfach ab

1.5 Bind

„Bind“ heißt: sich am LDAP-Server mit DN und Passwort anmelden. Gelingt der Bind, stimmt das Passwort. Genau so prüft WAIT Passwörter.

2. Was WAIT beim Login genau macht

Das steht im Code in `wait-backend/app/routers/auth.py` (Funktion `login`) und in `wait-backend/app/ldap_auth.py` (Funktion `authenticate`). Beispiel: Max tippt `m.muster` und sein Passwort ein.

Schritt 1: Sperre prüfen.

Gab es von dieser IP für diesen Namen in den letzten 15 Minuten 5 Fehlversuche, antwortet WAIT mit 429 „Too many failed attempts“, ohne den DC überhaupt zu fragen.

Schritt 2: Eingabe vorbereiten.

`SCHULE\m.muster` wird zu `m.muster`. Ist das Passwort leer, bricht WAIT sofort ab. Grund: AD wertet einen Bind mit Benutzername und *leerem* Passwort als „anonym erfolgreich“. Ohne diese Prüfung könnte man sich ohne Passwort anmelden.

Schritt 3: TLS-Verbindung zum DC aufbauen.

WAIT verbindet sich mit `LDAP_URL` und prüft dabei zwei Dinge:

1. Wurde das Zertifikat des DC von einer CA ausgestellt, die in `LDAP_CA_CERT_FILE` steht (bzw. einer öffentlich bekannten CA)?
2. Passt der **Hostname aus `LDAP_URL`** zum Namen im Zertifikat? (Siehe Kapitel 6.2, das ist die häufigste Falle.)

Ist eines davon falsch, bricht die Verbindung ab, und WAIT meldet „directory unreachable“.

Schritt 4: Als Dienstkonto anmelden.

WAIT macht einen Bind mit `LDAP_BIND_DN` und `LDAP_BIND_PASSWORD`. Das Dienstkonto ist nötig, weil AD anonyme Suchen nicht erlaubt.

Schritt 5: Benutzer suchen.

Unterhalb von `LDAP_BASE_DN` (inkl. aller Unterordner) sucht WAIT mit diesem Filter:

```
(&(objectCategory=person)(objectClass=user)
  (|(sAMAccountName=m.muster)(userPrincipalName=m.muster)(mail=m.muster)))
```

Auf Deutsch: „Eine Person, die ein Benutzer ist, und deren Anmeldename **oder** UPN **oder** E-Mail gleich `m.muster` ist.“

- **0 Treffer**: falscher Name, oder der Benutzer liegt außerhalb von `LDAP_BASE_DN`. Ergebnis: 401.
- **2 oder mehr Treffer** (z. B. zwei Konten mit derselben E-Mail): WAIT lehnt aus Sicherheitsgründen ab. Ergebnis: 401.
- **Genau 1 Treffer**: weiter.

Schritt 6: Passwort prüfen.

WAIT macht einen **zweiten** Bind, diesmal mit dem DN von Max und dem eingetippten Passwort.

- Bind klappt: Passwort richtig.
- Bind klappt nicht: Passwort falsch, **oder** das Konto ist deaktiviert, gesperrt oder abgelaufen, **oder** das Passwort ist abgelaufen bzw. muss geändert werden. WAIT kann diese Fälle nicht unterscheiden und meldet immer 401 „Incorrect email or password“.

Schritt 7: Gruppen prüfen.

Mit dem Dienstkonto fragt WAIT:

- Ist Max (direkt oder über verschachtelte Gruppen) in einer der Gruppen aus `LDAP_ADMIN_GROUP`? Wenn ja, wird er **Admin**.
- Ist `LDAP_USER_GROUP` gesetzt: Ist Max in einer dieser Gruppen (oder Admin)? Wenn nein, kommt 403 „Your account isn't allowed to use WAIT“.

„Verschachtelt“ heißt: Ist `WAIT-Admins` Mitglied in `IT-Team` und Max in `IT-Team`, dann zählt Max als Mitglied von `WAIT-Admins`. Technisch nutzt WAIT dafür die AD-Regel `LDAP_MATCHING_RULE_IN_CHAIN` (OID `1.2.840.113556.1.4.1941`).

Schritt 8: WAIT-Konto anlegen oder aktualisieren.

- WAIT liest `sAMAccountName`, `mail`, `userPrincipalName`, `displayName`, `givenName`, `sn`.
- **E-Mail für WAIT**: `mail`, sonst UPN, sonst `<sAMAccountName>@<LDAP_EMAIL_DOMAIN>`. Diese Adresse ist der interne Schlüssel des Kontos in WAIT und wird klein geschrieben.
- **Anzeigename**: `displayName`, sonst „Vorname Nachname“, sonst der Anmeldename.
- Beim **ersten** Login wird das Konto angelegt: `auth_source = ldap`, Sprache Deutsch, ein zufälliges lokales Passwort, das nie benutzt wird.

- Bei **jedem** Login werden Name und Rolle neu gesetzt:
  - `directory_admin` = „ist in AD-Admin-Gruppe“
  - Rolle = Admin, wenn `directory_admin` **oder** `panel_admin` (im WAIT-Panel zum Admin gemacht), sonst „customer“.

Schritt 9: Token ausgeben.

WAIT gibt ein JWT zurück (gültig `JWT_EXPIRE_MINUTES`, Standard 1440 = 24 h). Ab jetzt ist Max eingeloggt, ohne dass der DC erneut gefragt wird, bis das Token abläuft.

2.1 Die drei Modi (`AUTH_MODE`)

Modus	Wer kann sich anmelden	Registrierung	Passwort ändern
`local` (Standard)	nur WAIT-eigene Konten	erlaubt	erlaubt
`ldap`	nur AD-Konten	**aus**	**aus**
`both`	erst AD, dann WAIT-eigene Konten (z. B. `admin@wait.dev`)	**aus**	nur für lokale Konten

Bei `both`:

- Ist der DC **erreichbar** und das AD-Passwort falsch, prüft WAIT noch die lokalen Konten.
- Ist der DC **nicht erreichbar** und kein lokales Konto passt, kommt 503 „The school directory is unreachable“ statt „falsches Passwort“. So weiß der Benutzer, dass es nicht an ihm liegt.
- Ein AD-Benutzer kann sich **nicht** mit einem lokalen WAIT-Passwort anmelden, auch wenn eins existiert.

Das Login-Formular passt sich automatisch an: Bei `ldap` oder `both` steht dort „Benutzername oder E-Mail“, und der Link „Konto erstellen“ verschwindet.

Empfehlung: Starte mit `both`. Dann bleibt der lokale Admin `admin@wait.dev` als Notzugang, falls mit dem AD etwas nicht stimmt.

3. Vorbereitung auf dem Domänencontroller

Alle Schritte in diesem Kapitel machst du **auf dem DC** (oder einem PC mit RSAT-Tools), angemeldet als **Domänen-Admin**. PowerShell bitte „Als Administrator ausführen“.

3.1 Domäne und DC herausfinden

```
Get-ADDomain | Select-Object DNSRoot, NetBIOSName, DistinguishedName
Get-ADDomainController -Filter * | Select-Object HostName, IPv4Address, Site
```

Beispielausgabe:

```
DNSRoot      NetBIOSName DistinguishedName
-------      ----------- -----------------
schule.local SCHULE      DC=schule,DC=local

HostName           IPv4Address   Site
--------           -----------   ----
dc01.schule.local  192.168.92.10 Default-First-Site-Name
```

Trag die Werte in die Tabelle oben ein. Gibt es mehrere DCs, nimm einen davon (am besten den, der im selben Netz wie der WAIT-Server steht).

3.2 Dienstkonto anlegen

Warum? WAIT muss Benutzer suchen dürfen, bevor es deren Passwort prüft. Dafür braucht es ein eigenes Konto. Dieses Konto braucht **keine besonderen Rechte**: Jeder normale Domänenbenutzer darf das AD lesen.

a) Ordner (OU) für Dienstkonten anlegen (nur falls noch keiner existiert):

```
New-ADOrganizationalUnit -Name "Service" -Path "DC=schule,DC=local" -ProtectedFromAccidentalDeletion
$true
```

b) Ein starkes Passwort erzeugen (wird gleich gebraucht und später in die `.env` eingetragen):

```
-join ((48..57)+(65..90)+(97..122) | Get-Random -Count 32 | ForEach-Object {[char]$_})
```

Kopiere die Ausgabe und bewahre sie sicher auf (z. B. im Passwortmanager).

c) Konto anlegen:

```
New-ADUser -Name "svc-wait" `
  -SamAccountName "svc-wait" `
  -UserPrincipalName "svc-wait@schule.local" `
  -Description "Lesekonto für WAIT-Login (nicht löschen)" `
  -Path "OU=Service,DC=schule,DC=local" `
  -AccountPassword (Read-Host -AsSecureString "Passwort für svc-wait") `
  -Enabled $true `
  -PasswordNeverExpires $true `
  -CannotChangePassword $true
```

Bei „Passwort für svc-wait“ fügst du das Passwort aus Schritt b ein.

- `PasswordNeverExpires`: Liefe das Passwort ab, könnte sich plötzlich **niemand** mehr bei WAIT anmelden.
- `CannotChangePassword`: Niemand ändert es aus Versehen.

Per GUI (falls dir das lieber ist):

1. `dsa.msc` öffnen („Active Directory-Benutzer und -Computer“).
2. Rechtsklick auf die OU `Service` → **Neu** → **Benutzer**.
3. Vorname/Anmeldename `svc-wait` → **Weiter**.
4. Passwort eingeben, Haken bei **„Benutzer kann Kennwort nicht ändern“** und **„Kennwort läuft nie ab“**, Haken bei „Benutzer muss Kennwort bei der nächsten Anmeldung ändern“ **entfernen** → **Weiter** → **Fertig stellen**.

Wichtig: Trage beim Dienstkonto unter Eigenschaften → Konto → **Anmelden an…** keine Computer ein. Eine solche Beschränkung kann auch LDAP-Anmeldungen blockieren (AD-Fehler `data 531`). Dann käme niemand mehr bei WAIT rein.

3.3 Admin-Gruppe anlegen

Mitglieder dieser Gruppe werden in WAIT automatisch Admins (sehen das Admin-Panel und alle Tickets).

```
# Ordner für Gruppen (nur falls nicht vorhanden)
New-ADOrganizationalUnit -Name "Gruppen" -Path "DC=schule,DC=local"

# Gruppe anlegen
New-ADGroup -Name "WAIT-Admins" `
  -GroupScope Global -GroupCategory Security `
  -Description "Admins im WAIT-Helpdesk" `
  -Path "OU=Gruppen,DC=schule,DC=local"

# IT-Mitarbeiter hinzufügen
Add-ADGroupMember -Identity "WAIT-Admins" -Members "a.admin", "b.technik"
```

Mitglieder prüfen:

```
Get-ADGroupMember -Identity "WAIT-Admins" -Recursive | Select-Object SamAccountName, Name
```

Mehrere Admin-Gruppen? Geht. Du trägst sie später mit `;` getrennt ein (Kapitel 6). Eine bestehende Gruppe wie `IT-Team` kannst du auch direkt nutzen.

3.4 Optional: Gruppe für erlaubte Benutzer

Standardmäßig darf sich **jedes** aktive AD-Konto bei WAIT anmelden. Willst du das einschränken (z. B. nur Schüler und Lehrer, keine Dienst- oder Gastkonten), lege eine Gruppe an und nimm die passenden Gruppen als Mitglieder auf:

```
New-ADGroup -Name "WAIT-Benutzer" -GroupScope Global -GroupCategory Security -Path
"OU=Gruppen,DC=schule,DC=local"
Add-ADGroupMember -Identity "WAIT-Benutzer" -Members "Schueler", "Lehrer"   # bestehende Gruppen
```

Da verschachtelte Gruppen zählen, reicht es, die Gruppen `Schueler` und `Lehrer` hinzuzufügen.

3.5 Alle Werte für WAIT auslesen

Kopiere diesen Block komplett in PowerShell. Er gibt alle Werte aus, die du später in die `.env` einträgst:

```
$d = Get-ADDomain
"LDAP_BASE_DN=$($d.DistinguishedName)"
"LDAP_BIND_DN=$((Get-ADUser svc-wait).DistinguishedName)"
"LDAP_ADMIN_GROUP=$((Get-ADGroup WAIT-Admins).DistinguishedName)"
"LDAP_EMAIL_DOMAIN=$($d.DNSRoot)"
Get-ADDomainController -Filter * | ForEach-Object { "DC: $($_.HostName)  IP: $($_.IPv4Address)" }
# nur falls du 3.4 gemacht hast:
"LDAP_USER_GROUP=$((Get-ADGroup WAIT-Benutzer -ErrorAction SilentlyContinue).DistinguishedName)"
```

DN per GUI auslesen: `dsa.msc` → Menü **Ansicht** → **Erweiterte Features** aktivieren → Objekt öffnen → Reiter **Attribut-Editor** → `distinguishedName` doppelklicken → kopieren.

3.6 Einen Testbenutzer prüfen

Nimm einen echten Schüler- oder Lehreraccount, mit dem du später testest:

```
Get-ADUser m.muster -Properties mail, displayName, userPrincipalName, Enabled, LockedOut,
PasswordExpired, MemberOf |
  Select-Object DistinguishedName, SamAccountName, UserPrincipalName, mail, displayName, Enabled,
LockedOut, PasswordExpired
```

Achte auf:

- `DistinguishedName` endet auf deinen `LDAP_BASE_DN` (sonst findet WAIT ihn nicht).
- `Enabled = True`, `LockedOut = False`, `PasswordExpired = False` (sonst scheitert der Login mit „falsches Passwort“).

4. LDAPS (verschlüsselte Verbindung) einrichten

4.1 Prüfen, ob LDAPS schon läuft

Oft ist LDAPS schon aktiv, ohne dass es jemand weiß, nämlich wenn die Domäne eine Unternehmens-CA hat. Prüfe das zuerst.

Test 1: Port offen? (auf dem DC oder einem Windows-PC in der Domäne)

```
Test-NetConnection dc01.schule.local -Port 636
```

`TcpTestSucceeded : True` heißt nur, dass der Port offen ist, noch nicht, dass TLS funktioniert.

Test 2: TLS funktioniert? Mit `ldp.exe`:

1. Auf dem DC `ldp.exe` starten (Win+R → `ldp`).
2. **Verbindung** → **Verbinden…**
3. Server: `dc01.schule.local` (der **Name**, nicht die IP), Port: `636`, Haken bei **SSL** → **OK**.
4. Erscheint rechts ein langer Text mit `Established connection to dc01.schule.local` und RootDSE-Infos (`defaultNamingContext` usw.): **LDAPS läuft.** Weiter mit 4.4.
5. Kommt `Error <0x51>: Fail to connect` o. ä.: LDAPS läuft nicht. Weiter mit 4.2.

4.2 Gibt es eine Zertifizierungsstelle (CA)?

```
certutil -config - -ping
```

oder einfacher:

```
certutil -dump | Select-String "Config"
```

- Wird eine CA angezeigt (z. B. `dc01.schule.local\schule-DC01-CA`): weiter mit **4.3 Fall A**.
- Fehlermeldung / keine CA: weiter mit **4.3 Fall B**.

Alternativ: In `certlm.msc` → **Vertrauenswürdige Stammzertifizierungsstellen** → **Zertifikate** nach einem Eintrag wie `schule-DC01-CA` suchen.

4.3 Zertifikat für den DC besorgen

Fall A: Es gibt eine Unternehmens-CA

1. Auf dem DC Win+R → `certlm.msc` („Zertifikate – Lokaler Computer“).
2. Links **Eigene Zertifikate** → **Zertifikate** öffnen.
3. Schau zuerst, ob dort schon ein Zertifikat mit Zweck **„Serverauthentifizierung“** und **Ausgestellt für** = `dc01.schule.local` liegt. Wenn ja: Es fehlt evtl. nur ein Neustart. Mach 4.1 Test 2 nach einem Neustart erneut.
4. Wenn nein: Rechtsklick auf **Zertifikate** → **Alle Aufgaben** → **Neues Zertifikat anfordern…**
5. **Weiter** → **Active Directory-Registrierungsrichtlinie** → **Weiter**.
6. Haken bei **Domänencontroller** (oder „Kerberos-Authentifizierung“, beide sind passend) → **Registrieren** → **Fertig stellen**.
7. **DC neu starten** (oder siehe Hinweis unten), dann 4.1 Test 2 wiederholen.

Hinweis: Microsoft beschreibt, dass das AD ein Zertifikat auch ohne Neustart übernehmen kann, wenn man es in den Zertifikatspeicher des NTDS-Dienstes legt. Ein Neustart des DC ist aber der einfachste und sicherste Weg.

Fall B: Es gibt keine CA, also eine einrichten

In kleinen Schulnetzen wird die CA oft direkt auf dem DC installiert. Microsoft empfiehlt eigentlich einen eigenen Server, für eine Schule ist der DC aber üblich und funktioniert.

1. **Server-Manager** → **Verwalten** → **Rollen und Features hinzufügen**.
2. **Weiter** bis „Serverrollen“ → Haken bei **Active Directory-Zertifikatdienste** → Features hinzufügen → **Weiter**.
3. Bei „Rollendienste“ nur **Zertifizierungsstelle** anhaken → **Weiter** → **Installieren**.
4. Nach der Installation oben im Server-Manager auf die gelbe Flagge → **Active Directory-Zertifikatdienste auf dem Zielserver konfigurieren**.
5. Anmeldeinformationen: Domänen-Admin → **Weiter**.

6. Rollendienste: **Zertifizierungsstelle** → **Weiter**.
7. Setuptyp: **Unternehmenszertifizierungsstelle** → **Weiter**.
8. ZS-Typ: **Stammzertifizierungsstelle** → **Weiter**.
9. Privater Schlüssel: **Neuen privaten Schlüssel erstellen** → **Weiter**.
10. Kryptografie: RSA, **4096** Bit, **SHA256** → **Weiter**.
11. Name: Vorschlag übernehmen (z. B. `schule-DC01-CA`) → **Weiter**.
12. Gültigkeitsdauer: z. B. **10 Jahre** → **Weiter** → **Konfigurieren**.
13. Danach holt sich der DC das Domänencontroller-Zertifikat meist automatisch (automatische Registrierung). Zur Sicherheit: `gpupdate /force` und dann Fall A ab Schritt 1.
14. **DC neu starten**, dann 4.1 Test 2.

4.4 Zertifikat des DC kontrollieren

In `certlm.msc` → **Eigene Zertifikate** → **Zertifikate** → das DC-Zertifikat doppelklicken:

Reiter	Was dort stehen muss
Allgemein	„Sie besitzen einen privaten Schlüssel für dieses Zertifikat“
Details → **Erweiterte Schlüsselverwendung**	enthält **Serverauthentifizierung (1.3.6.1.5.5.7.3.1)**
Details → **Alternativer Antragstellername**	enthält `DNS-Name=dc01.schule.local`
Zertifizierungspfad	„Dieses Zertifikat ist gültig“

Merk dir den DNS-Namen aus „Alternativer Antragstellername“. **Genau diesen Namen** brauchst du in Kapitel 6 für `LDAP_URL`.

4.5 Das CA-Zertifikat für WAIT exportieren

WAIT (Linux) kennt deine schulinterne CA nicht. Du gibst ihm deshalb das **öffentliche** CA-Zertifikat. Es ist kein Geheimnis.

Auf dem Server mit der CA (oft der DC):

```
certutil -ca.cert C:\ca.cer
certutil -encode C:\ca.cer C:\ca.pem
notepad C:\ca.pem
```

Der Inhalt muss so aussehen:

```
-----BEGIN CERTIFICATE-----
MIIF... (viele Zeilen Base64)
-----END CERTIFICATE-----
```

Alternative, falls `certutil -ca.cert` nicht geht (z. B. weil du nicht auf dem CA-Server bist): `certlm.msc` → **Vertrauenswürdige Stammzertifizierungsstellen** → **Zertifikate** → deine CA (z. B. `schule-DC01-CA`) → Rechtsklick → **Alle Aufgaben** → **Exportieren…** → Format **„Base-64-codiert X.509 (.CER)“** → als `ca.pem` speichern.

Gibt es eine Zwischen-CA (Zertifizierungspfad im DC-Zertifikat hat 3 Ebenen statt 2)? Dann exportiere Stamm- **und** Zwischenzertifikat und füge beide Blöcke hintereinander in dieselbe `ca.pem` ein.

4.6 Firewall

Der WAIT-Server muss den DC auf **TCP 636** erreichen.

- Die Windows-Firewall auf dem DC lässt 636 bei installiertem AD normalerweise schon zu.

- Liegt eine weitere Firewall (z. B. Sophos, OPNsense, pfSense) zwischen WAIT-Server und DC: Regel **WAIT-Server-IP → DC-IP, TCP 636, erlauben** anlegen.

5. WAIT-Server vorbereiten

Alle Befehle auf dem WAIT-Server (Debian) als Deploy-Benutzer, im Projektordner (laut `DEPLOYMENT.md` `~/wait`).

5.1 Werkzeuge zum Testen installieren

```
sudo apt-get install -y netcat-openbsd ldap-utils openssl
```

5.2 Netzwerk zum DC prüfen

```
nc -zv 192.168.92.10 636
```

- `Connection to 192.168.92.10 636 port [tcp/ldaps] succeeded!`: gut.
- `Connection refused`: LDAPS läuft auf dem DC nicht (zurück zu Kapitel 4).
- `timed out`: Eine Firewall blockiert (Kapitel 4.6) oder die IP ist falsch.

Trag den DC-Namen danach auf dem WAIT-Server ein. Die Tests in Kapitel 5.4 und 7 prüfen den Namen im Zertifikat und brauchen deshalb den Namen statt der IP:

```
getent hosts dc01.schule.local || echo "192.168.92.10 dc01.schule.local" | sudo tee -a /etc/hosts
```

5.3 CA-Zertifikat auf den Server kopieren

Von deinem Windows-PC aus (PowerShell), wenn `ca.pem` z. B. in deinem Download-Ordner liegt:

```
scp -i $HOME\.ssh\wait_deploy_ed25519 $HOME\Downloads\ca.pem
deploy@192.168.92.100:~/wait/certs/ca.pem
```

Oder auf dem Server mit `nano ~/wait/certs/ca.pem` öffnen und den Inhalt einfügen.

Prüfen:

```
openssl x509 -in ~/wait/certs/ca.pem -noout -subject -enddate
```

Ausgabe z. B. `subject=DC=local, DC=schule, CN=schule-DC01-CA` und ein Ablaufdatum. Kommt `unable to load certificate`, ist die Datei kaputt (meist fehlt die BEGIN/END-Zeile oder es ist das binäre `.cer`).

5.4 Zertifikat des DC vom WAIT-Server aus prüfen

```
openssl s_client -connect dc01.schule.local:636 -CAfile ~/wait/certs/ca.pem </dev/null 2>/dev/null \
  | openssl x509 -noout -subject -ext subjectAltName
openssl s_client -connect dc01.schule.local:636 -CAfile ~/wait/certs/ca.pem \
  -verify_hostname dc01.schule.local </dev/null 2>/dev/null | grep "Verify return code"
```

- Die erste Ausgabe zeigt, auf welchen Namen das Zertifikat lautet, z. B. `DNS:dc01.schule.local`.
- `Verify return code: 0 (ok)` heißt: `ca.pem` passt zum DC-Zertifikat, und der Name stimmt.
- `unable to get local issuer certificate (20)` heißt: falsche oder unvollständige `ca.pem` (evtl. fehlt die Zwischen-CA, siehe 4.5).
- `hostname mismatch (62)` heißt: Das Zertifikat lautet auf einen anderen Namen. Nimm den Namen aus der ersten Ausgabe.

6. WAIT konfigurieren

6.1 Die Einstellungen im Einzelnen

Variable	Pflicht	Bedeutung	Beispiel
AUTH_MODE	ja	local, ldap oder both (Kapitel 2.1)	both
LDAP_URL	ja	Adresse des DC. **Der Hostname muss im DC-Zertifikat stehen** (6.2)	ldaps://dc01.schule.local:636
LDAP_BASE_DN	ja	Ab hier wird nach Benutzern gesucht (inkl. Unterordner)	DC=schule,DC=local
LDAP_BIND_DN	ja	Kompletter DN des Dienstkontos	CN=svc-wait,OU=Service,DC=schule,DC=local
LDAP_BIND_PASSWORD	ja	Passwort des Dienstkontos	(geheim)
LDAP_ADMIN_GROUP	empfohlen	DN(s) der Admin-Gruppe(n), mehrere mit ; getrennt	CN=WAIT-Admins,OU=Gruppen,DC=schule,DC=local
LDAP_USER_GROUP	nein	Wenn gesetzt: nur diese Gruppe(n) (und Admins) dürfen rein. Leer = alle	leer
LDAP_EMAIL_DOMAIN	nein	Domäne für Konten ohne mail/UPN. Leer = wird aus LDAP_BASE_DN gebildet	schule.local
LDAP_CA_CERT_FILE	bei interner CA	Pfad **im Container** zur ca.pem	/certs/ca.pem
LDAP_START_TLS	nein	Nur für ldap://…:389 + StartTLS. Bei ldaps:// immer false	false
LDAP_TIMEOUT_SECONDS	nein	Wie lange WAIT auf den DC wartet (ganze Sekunden)	5

Hinweise:

- **Mehrere Gruppen werden mit ; getrennt**, nicht mit Komma, weil DNs selbst Kommas enthalten.
- Den Suchfilter (LDAP_USER_FILTER) kannst du im Normalfall so lassen. Er steht in wait-backend/app/config.py und wird von docker-compose.yml derzeit nicht durchgereicht.

6.2 Wichtig: Hostname statt IP in LDAP_URL

Das README und .env.example schlagen ldaps://<DC-IP>:636 vor, weil Docker-Container interne Namen wie dc01.schule.local oft nicht auflösen können.

Aber: Die Bibliothek ldap3, mit der WAIT den DC anspricht, prüft beim Verbindungsaufbau, ob der Name aus LDAP_URL im Zertifikat steht (Funktion check_hostname in ldap3/core/tls.py, Version 2.9.1, nachgelesen). WAIT übergibt dabei keine Ausnahmen. Ein normales Domänencontroller-Zertifikat enthält nur **DNS-Namen**, keine IP. Mit der IP in der URL bricht die Verbindung deshalb mit „certificate … doesn't match any name“ ab, und WAIT meldet 503.

Ich habe das am Quellcode geprüft, aber nicht live gegen deinen DC getestet. Falls es bei dir mit IP schon funktioniert, steht die IP im Zertifikat, und du kannst bei der IP bleiben.

Lösung (empfohlen): FQDN in die URL und dem Container den Namen beibringen.

1. In der .env:

```
LDAP_URL=ldaps://dc01.schule.local:636
```

2. In `docker-compose.yml` beim Dienst `backend` einen festen Hosteintrag ergänzen (auf gleicher Ebene wie `volumes:` und `ports:`):

```
  backend:
    build: ./wait-backend
    # ... bestehende Einträge ...
    extra_hosts:
      - "dc01.schule.local:192.168.92.10"
```

Damit löst der Container `dc01.schule.local` immer zu `192.168.92.10` auf, ganz ohne DNS.

Wie bei der `CORS_ORIGINS`-Änderung aus `DEPLOYMENT.md` ist das eine lokale Änderung an `docker-compose.yml`. Bei `git pull` musst du sie per `git stash` sichern und danach wieder anwenden (oder du nimmst sie dauerhaft ins Repo auf).

Alternative: Beim Ausstellen des DC-Zertifikats die IP als zusätzlichen SAN-Eintrag aufnehmen (`IP-Adresse=192.168.92.10`). Das ist aufwendiger (eigene Zertifikatvorlage oder `certreq` mit `.inf`-Datei) und lohnt sich meist nicht.

6.3 Die `.env` schreiben

Bei Docker liegt die `.env` im Projekt-Root `~/wait/.env`, also neben `docker-compose.yml`. Dort stehen schon `POSTGRES_PASSWORD`, `JWT_SECRET` usw. aus `DEPLOYMENT.md`. **Nicht überschreiben, sondern ergänzen:**

```
cd ~/wait
cp .env .env.backup
nano .env
```

Unten anhängen (Werte aus Kapitel 3.5):

```
# ---- Active Directory ----
AUTH_MODE=both
LDAP_URL=ldaps://dc01.schule.local:636
LDAP_BASE_DN=DC=schule,DC=local
LDAP_BIND_DN=CN=svc-wait,OU=Service,DC=schule,DC=local
LDAP_BIND_PASSWORD=HierDasPasswortAus3.2
LDAP_ADMIN_GROUP=CN=WAIT-Admins,OU=Gruppen,DC=schule,DC=local
LDAP_USER_GROUP=
LDAP_EMAIL_DOMAIN=schule.local
LDAP_CA_CERT_FILE=/certs/ca.pem
LDAP_START_TLS=false
```

Speichern mit Strg+O, Enter, Strg+X. Danach:

```
chmod 600 .env
```

Fallen bei der `.env`:

- **Keine Leerzeichen** um das `=`.
- **Keine Anführungszeichen** nötig, auch nicht bei DNs mit Kommas.
- Enthält das Passwort ein `$`, interpretiert Docker Compose das als Variable. Entweder ein Passwort ohne `$` wählen oder jedes `$` als `$$` schreiben.
- `LDAP_CA_CERT_FILE` ist der Pfad **im Container** (`/certs/ca.pem`), nicht auf dem Server (`~/wait/certs/ca.pem`). `docker-compose.yml` verbindet die beiden mit `./certs:/certs:ro`.

`LDAP_TIMEOUT_SECONDS` wird von `docker-compose.yml` nicht durchgereicht, es bleibt beim Standard 5 Sekunden. Das passt normalerweise.

Ohne Docker (lokale Entwicklung): Dieselben Zeilen kommen in `wait-backend/.env`, und `LDAP_CA_CERT_FILE` ist dann ein normaler Dateipfad, z. B. `../certs/ca.pem`.

6.4 Prüfen, ob Compose die Werte sieht

```
docker compose config | grep -E "AUTH_MODE|LDAP_"
```

Alle Werte müssen ausgefüllt erscheinen. Steht dort `AUTH_MODE: local` oder leere Werte, liest Compose deine `.env` nicht (falscher Ordner oder Tippfehler im Variablennamen).

6.5 Neu starten

```
docker compose up -d --build backend
docker compose restart frontend
```

Das `restart frontend` ist wichtig: Laut `DEPLOYMENT.md` merkt sich nginx sonst die alte IP des Backend-Containers, und jede Anfrage endet mit 502.

Prüfen, ob die Werte im Container angekommen sind und das Zertifikat da ist:

```
docker compose exec backend sh -c 'env | grep -E "AUTH_MODE|LDAP_" | grep -v PASSWORD; ls -l /certs;
getent hosts dc01.schule.local'
```

Erwartet: Die LDAP-Werte, `ca.pem` in `/certs`, und `192.168.92.10 dc01.schule.local`.

7. Testen, Schritt für Schritt

Mach die Tests in dieser Reihenfolge. Jeder baut auf dem vorherigen auf, so siehst du genau, wo es hakt.

Test 1: Dienstkonto + Zertifikat + Suche (ohne WAIT)

Auf dem WAIT-Server:

```
LDAPTLS_CACERT=~/wait/certs/ca.pem ldapsearch -LLL \
  -H ldaps://dc01.schule.local:636 \
  -D "CN=svc-wait,OU=Service,DC=schule,DC=local" -W \
  -b "DC=schule,DC=local" \
  "(sAMAccountName=m.muster)" \
  sAMAccountName userPrincipalName mail displayName memberOf
```

`-W` fragt nach dem Passwort des Dienstkontos.

Hinweis: `ldapsearch` prüft wie WAIT den Namen im Zertifikat. Deshalb steht hier der DC-Name und nicht die IP. Klappt dieser Test, WAIT aber nicht, prüfe `extra_hosts` und die `.env` (Kapitel 6).

Ergebnis	Bedeutung
Ein Eintrag `dn: CN=Max Muster,...` mit Attributen	Dienstkonto, Zertifikat und Base-DN stimmen.
`ldap_bind: Invalid credentials (49)`	`LDAP_BIND_DN` oder Passwort falsch.
`Can't contact LDAP server (-1)`	Netzwerk/Firewall oder Zertifikat. `openssl`-Test aus 5.4 machen.
Kein Ergebnis, aber kein Fehler	Benutzer existiert nicht unter dieser Base-DN oder der Name stimmt nicht.

Test 2: Passwort eines echten Benutzers

```
LDAPTLS_CACERT=~/wait/certs/ca.pem ldapwhoami \
  -H ldaps://dc01.schule.local:636 \
  -D "CN=Max Muster,OU=Schueler,DC=schule,DC=local" -W
```

(Den DN aus Test 1 kopieren.) `u:SCHULE\m.muster` heißt: Das Passwort stimmt. Das ist genau der Bind, den WAIT in Schritt 6 macht.

Test 3: Gruppenprüfung (verschachtelt)

```
LDAPTLS_CACERT=~/wait/certs/ca.pem ldapsearch -LLL \
  -H ldaps://dc01.schule.local:636 \
  -D "CN=svc-wait,OU=Service,DC=schule,DC=local" -W \
  -b "CN=A Admin,OU=IT,DC=schule,DC=local" -s base \
  "(memberOf:1.2.840.113556.1.4.1941:=CN=WAIT-Admins,OU=Gruppen,DC=schule,DC=local)" dn
```

Ist der Benutzer (direkt oder verschachtelt) in `WAIT-Admins`, kommt sein DN zurück. Kommt nichts, wird er in WAIT **kein** Admin. Genau diese Abfrage macht WAIT.

Test 4: Health-Check von WAIT

```
wget -qO- http://localhost:8080/api/health
wget -qO- http://localhost:8080/api/auth/config
```

Erwartet: `{"status":"ok"}` und `{"mode":"both","registration_enabled":false}`.

Test 5: Login direkt über die API

```
curl -s -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"m.muster","password":"DAS-PASSWORT"}'
```

(Das Feld heißt `email`, nimmt aber auch Benutzernamen.)

Antwort	Bedeutung
`{"access_token":"eyJ...","token_type":"bearer","user":{...}}`	Login klappt.
`401 Incorrect email or password`	Passwort, Name oder Base-DN (siehe Tests 1 und 2).
`403 Your account isn't allowed to use WAIT`	`LDAP_USER_GROUP` gesetzt, Benutzer nicht drin.
`503 The school directory is unreachable`	Verbindung/Zertifikat/Dienstkonto. Log ansehen (Kapitel 8.1).

Danach das Passwort aus dem Terminal-Verlauf löschen: `history -c`.

Test 6: Im Browser

1. `http://192.168.92.100:8080` öffnen.
2. Das Formular zeigt jetzt „Benutzername oder E-Mail“ und keinen Link „Konto erstellen“ mehr.
3. Mit einem **normalen** AD-Benutzer anmelden: Er landet in der Ticketübersicht.
4. Abmelden, mit einem Benutzer aus `WAIT-Admins` anmelden: Er sieht das Admin-Panel (`/admin`).
5. Im Admin-Panel unter **Team** prüfen: Beide Konten sind angelegt, mit Namen aus dem AD.
6. Der lokale Notzugang `admin@wait.dev` funktioniert weiterhin (wegen `both`).

8. Fehlersuche

8.1 Das Backend-Log lesen

```
docker compose logs --tail=100 backend | grep -i ldap
```

Live mitlesen, während du dich anmeldest:

```
docker compose logs -f backend
```

Log-Meldung	Ursache	Lösung
`LDAP_CA_CERT_FILE /certs/ca.pem not found in the container (is it in ./certs?)`	`ca.pem` liegt nicht in `~/wait/certs/` oder der Container wurde vorher gestartet	Datei ablegen, `docker compose up -d backend`
`LDAP service account bind failed: invalidCredentials`	`LDAP_BIND_DN` oder `LDAP_BIND_PASSWORD` falsch, oder Dienstkonto deaktiviert/abgelaufen	DN neu auslesen (3.5), Passwort prüfen, `$` im Passwort (6.3)
`certificate ... doesn't match any name in [...]`	Hostname in `LDAP_URL` steht nicht im Zertifikat	Kapitel 6.2
`certificate verify failed` / `unable to get local issuer certificate`	`ca.pem` passt nicht oder Zwischen-CA fehlt	Kapitel 4.5 + Test 5.4
`socket connection error` / `timed out` / `Name or service not known`	Netzwerk, Firewall oder Container kann den Namen nicht auflösen	`nc -zv`, `extra_hosts` (6.2), `getent hosts` im Container (6.5)
keine LDAP-Meldung, aber 401	Verbindung ok, Benutzer nicht gefunden oder Passwort falsch	Tests 1 und 2

8.2 Benutzer bekommt „falsches Passwort“, obwohl es stimmt

Der Passwort-Bind scheitert auch, wenn das Konto nicht nutzbar ist. Auf dem DC prüfen:

```
Get-ADUser m.muster -Properties Enabled, LockedOut, PasswordExpired, AccountExpirationDate,
pwdLastSet |
  Select-Object Enabled, LockedOut, PasswordExpired, AccountExpirationDate, pwdLastSet
```

- `Enabled = False`: Konto deaktiviert.
- `LockedOut = True`: Konto gesperrt (zu viele Fehlversuche, auch von anderen Geräten). Entsperren: `Unlock-ADAccount m.muster`.
- `PasswordExpired = True` oder `pwdLastSet = 0` („muss Passwort bei nächster Anmeldung ändern“): Der Schüler muss sich **erst einmal an einem Schul-PC anmelden** und das Passwort ändern. WAIT kann Passwörter nicht ändern.
- `AccountExpirationDate` in der Vergangenheit: Konto abgelaufen.

Außerdem: Liegt der Benutzer außerhalb von `LDAP_BASE_DN`? Gibt es zwei Konten mit derselben `mail`?

```
Get-ADUser -Filter "mail -eq 'max.muster@schule.de'" | Select-Object DistinguishedName
```

8.3 Benutzer ist kein Admin, obwohl er in der Gruppe ist

- DN in `LDAP_ADMIN_GROUP` exakt prüfen (3.5). Ein falscher Gruppen-DN führt nicht zu einer Fehlermeldung, sondern einfach dazu, dass niemand Admin wird.
- Mehrere Gruppen mit `;` getrennt, nicht mit `,`.
- Gruppenänderungen im AD wirken erst beim **nächsten Login** in WAIT. Abmelden und neu anmelden.
- Test 3 aus Kapitel 7 durchführen.

8.4 Admin lässt sich im Panel nicht zum Kunden machen

Das ist Absicht: Wer über eine AD-Gruppe Admin ist, kann im Panel nicht herabgestuft werden. Die Meldung lautet „This account is an admin through an Active Directory group. Remove it from the group in AD.“ Lösung: Im AD aus `WAIT-Admins` entfernen, dann beim nächsten Login ist er Kunde.

8.5 Nach `docker compose up -d --build backend` geht gar nichts mehr (502)

`docker compose restart frontend` (siehe `DEPLOYMENT.md`, „nginx caches a stale backend IP“).

8.6 „Too many failed attempts“

Nach 5 Fehlversuchen sperrt WAIT diese Kombination aus IP und Name für 15 Minuten. Die Sperre liegt im Arbeitsspeicher des Backends. `docker compose restart backend` hebt sie sofort auf.

9. Betrieb: Was du danach wissen musst

9.1 Umstellen auf „nur AD“

Wenn alles läuft und alle IT-Leute als AD-Admins reinkommen:

```
sed -i 's/^AUTH_MODE=both/AUTH_MODE=ldap/' ~/wait/.env
docker compose up -d backend && docker compose restart frontend
```

Achtung: Danach funktioniert auch `admin@wait.dev` nicht mehr. Fällt der DC aus, kommt niemand mehr in WAIT. Viele lassen deshalb dauerhaft `both` und ändern nur das Passwort des Seed-Admins.

9.2 Seed-Konten absichern

`admin@wait.dev` / `admin-wait-2026` und `j.alvarez@company.com` stehen öffentlich im README. Bei `both` sind sie weiterhin aktiv.

- Melde dich mit beiden Konten an und ändere jeweils das Passwort unter „Passwort ändern“. WAIT hat keine Funktion, um Konten zu löschen.
- Setze danach `SEED_ON_START=0` in der `.env`, damit der Seed nicht erneut läuft.

9.3 AD-Kontosperre beachten

Jeder falsche Login in WAIT ist auch ein falscher Bind am DC und zählt in die **Kontosperrungsrichtlinie** des AD. Probiert jemand in WAIT mehrfach falsche Passwörter für `m.muster`, kann Max dadurch auch an den Schul-PCs gesperrt werden. Die WAIT-eigene Sperre (5 Versuche / 15 Minuten pro IP) bremst das zwar, verhindert es aber nicht über verschiedene IPs.

9.4 Was passiert, wenn …

Ereignis	Auswirkung in WAIT
Schüler verlässt die Schule, Konto wird deaktiviert	Kann sich nicht mehr anmelden. Ein bestehendes Token gilt bis zum Ablauf (max. 24 h) weiter. Das WAIT-Konto mit Tickets bleibt erhalten.
Name im AD geändert	Wird beim nächsten Login übernommen.
E-Mail im AD geändert	**Achtung:** Die E-Mail ist der Schlüssel des WAIT-Kontos. Bei neuer E-Mail legt WAIT ein **neues** Konto an, alte Tickets hängen am alten.
Aus `WAIT-Admins` entfernt	Beim nächsten Login Kunde (außer er wurde im Panel zum Admin gemacht).
DC-Zertifikat erneuert	Kein Problem, solange es von derselben CA kommt.
CA-Zertifikat erneuert/getauscht	Neue `ca.pem` exportieren (4.5), nach `certs/` kopieren, `docker compose restart backend`.
Passwort von `svc-wait` geändert	`.env` anpassen, `docker compose up -d backend`. Bis dahin meldet WAIT 503.
DC-IP ändert sich	`extra_hosts` in `docker-compose.yml` anpassen (bzw. `LDAP_URL`).

9.5 Ablaufdaten im Blick behalten

```
openssl x509 -in ~/wait/certs/ca.pem -noout -enddate
openssl s_client -connect dc01.schule.local:636 </dev/null 2>/dev/null | openssl x509 -noout -enddate
```

Das DC-Zertifikat aus der Vorlage „Domänencontroller“ wird bei einer Unternehmens-CA automatisch verlängert. Die CA selbst läuft je nach Einstellung nach 5 bis 10 Jahren ab.

9.6 Updates von WAIT

Vor `git pull` deine lokalen Änderungen an `docker-compose.yml` (`CORS_ORIGINS`, `extra_hosts`) sichern, wie in `DEPLOYMENT.md` Kapitel 8 beschrieben. `.env` und `certs/ca.pem` sind nicht im Git und bleiben bei `git pull` unverändert.

10. Checkliste

Auf dem DC

- ☐ Domäne, DC-Name und DC-IP notiert (3.1)
- ☐ Dienstkonto `svc-wait` angelegt, Passwort läuft nie ab (3.2)
- ☐ Gruppe `WAIT-Admins` angelegt, IT-Leute drin (3.3)
- ☐ Optional: Gruppe `WAIT-Benutzer` (3.4)
- ☐ Alle DNs ausgelesen (3.5)
- ☐ LDAPS läuft, `ldp.exe` mit SSL auf 636 klappt (4.1)
- ☐ DNS-Name im DC-Zertifikat notiert (4.4)
- ☐ `ca.pem` exportiert (4.5)
- ☐ Firewall: WAIT-Server → DC TCP 636 frei (4.6)

Auf dem WAIT-Server

- ☐ `nc -zv <DC-IP> 636` klappt (5.2)
- ☐ `certs/ca.pem` liegt da, `openssl x509` kann sie lesen (5.3)
- ☐ `openssl s_client` meldet `Verify return code: 0 (ok)` (5.4)
- ☐ `LDAP_URL` mit FQDN + `extra_hosts` in `docker-compose.yml` (6.2)
- ☐ `.env` ergänzt, `chmod 600` (6.3)
- ☐ `docker compose config` zeigt die Werte (6.4)
- ☐ Backend neu gebaut, Frontend neu gestartet (6.5)

Tests

- ☐ `ldapsearch` mit Dienstkonto findet Testbenutzer (Test 1)
- ☐ `ldapwhoami` mit Testbenutzer klappt (Test 2)
- ☐ Gruppenabfrage für Admin klappt (Test 3)
- ☐ `/api/auth/config` zeigt `mode: both` (Test 4)
- ☐ API-Login liefert Token (Test 5)
- ☐ Browser: normaler Benutzer = Kunde, `WAIT-Admins` = Admin (Test 6)

Danach

- ☐ Passwörter beider Seed-Konten geändert, `SEED_ON_START=0` (9.2)
- ☐ Entscheidung: bei `both` bleiben oder auf `ldap` umstellen (9.1)

Quellen

- WAIT-Repository `I3MC-eu/WAIT` (Branch `main`): `README.md`, `DEPLOYMENT.md`, `docker-compose.yml`, `wait-backend/.env.example`, `wait-backend/app/config.py`, `wait-backend/app/ldap_auth.py`, `wait-backend/app/routers/auth.py`, `wait-backend/app/routers/admin.py`
- `ldap3` 2.9.1, `ldap3/core/tls.py` (Hostnamen-Prüfung)
- Microsoft Learn: Configure certificates for LDAP over SSL in AD DS
- Microsoft Learn: Enable LDAP over SSL with a third-party certification authority
- Microsoft Learn: Troubleshoot LDAP over SSL connection problems
- Früherer Chat zu WAIT/AD (Dienstkonto, `certutil`-Export, `ldp.exe`-Test)